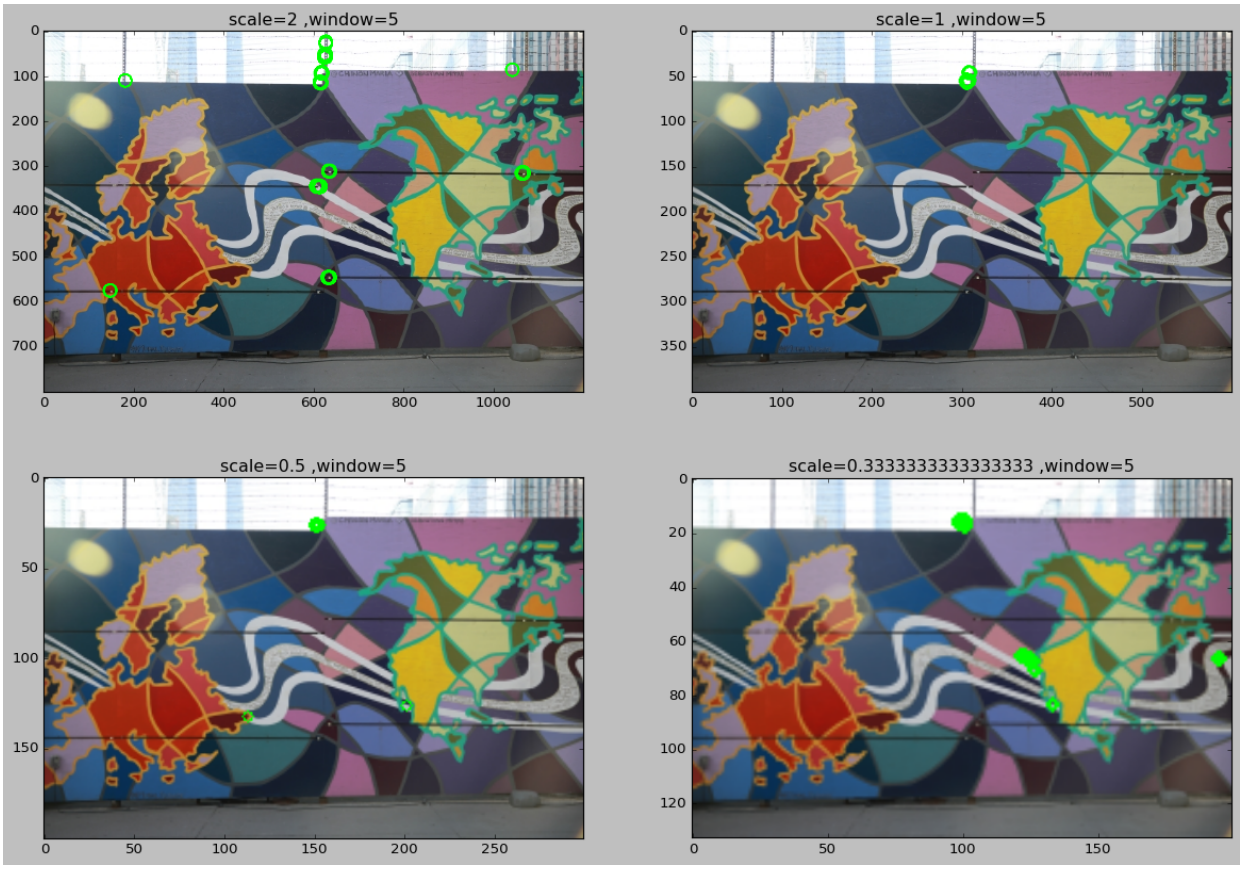
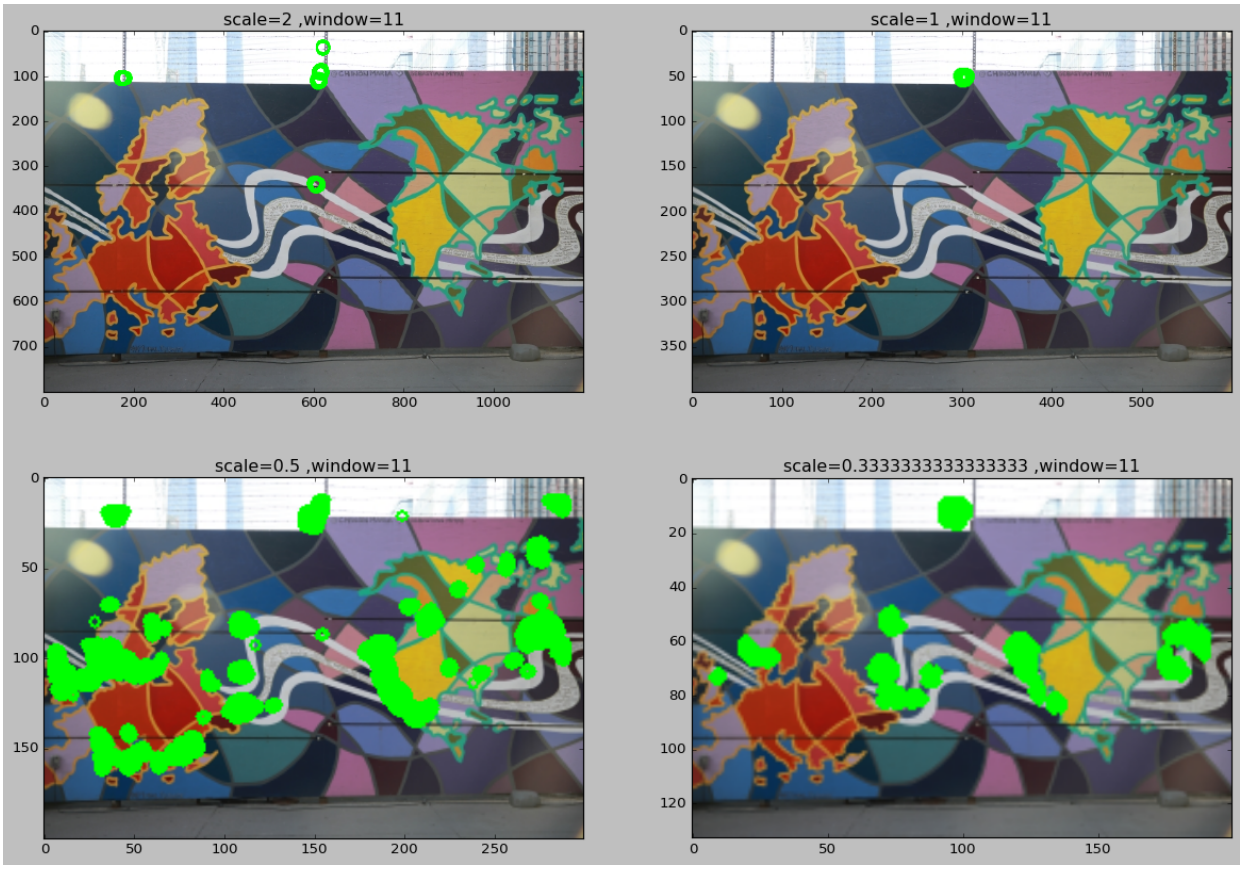
In [ ]:
```
from functions import *
import cv2
import numpy as np
import matplotlib.pyplot as plt
plt.style.use(plt.style.available[5])
```

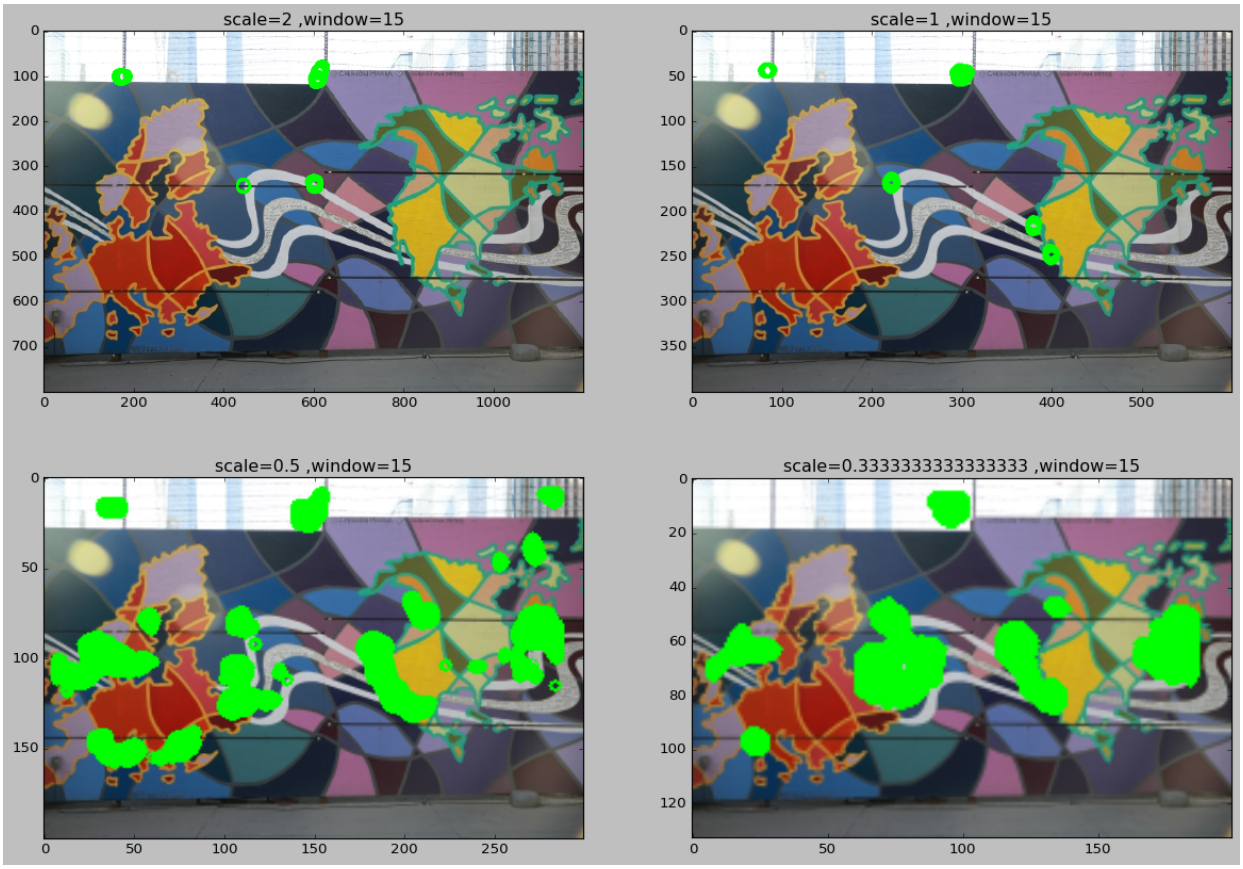
7.1. Harris Corner Detector

7.1.1. Extract interest points using the Harris Corner detector that you implemented. In this way, apply the Harris Corner detector for at least 4 different scales. Which interest points do you observe to be detected across all these different scales? Notice that your implementation should allow for any suitable scale as input, however you can show results on a minimum of 4 different scales (Test on harris.JPG Image).

In [ ]:
```
img = cv2.imread("harris.JPG")
harris_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

windows = [5,11,15]
scales = [2,1,1/2,1/3]

for window in windows:
    i = 0
    figure = plt.figure(figsize=(18,12))
    for scale in scales:
        copy = np.array(harris_img)
        # interpolation=3 -> INTER_AREA
        img_scaled = cv2.resize(copy, (None), fx=scale, fy=scale, interpolation=3)

        harris_with_corners = harris_corner_detector(img_scaled,window_size=window,thr

        figure.add_subplot(2,2,i+1)
        i +=1
        plt.imshow(harris_with_corners)
        plt.title(f'scale={str(scale)} ,window={str(window)}')
    plt.show()
```


scale=2 ,window=5
scale=1 ,window=5
scale=0.5 ,window=5
scale=0.3333333333333333 ,window=5


scale=2 ,window=11
scale=1 ,window=11
scale=0.5 ,window=11
scale=0.3333333333333333 ,window=11



7.2. Scene stitching with SIFT/SURF features

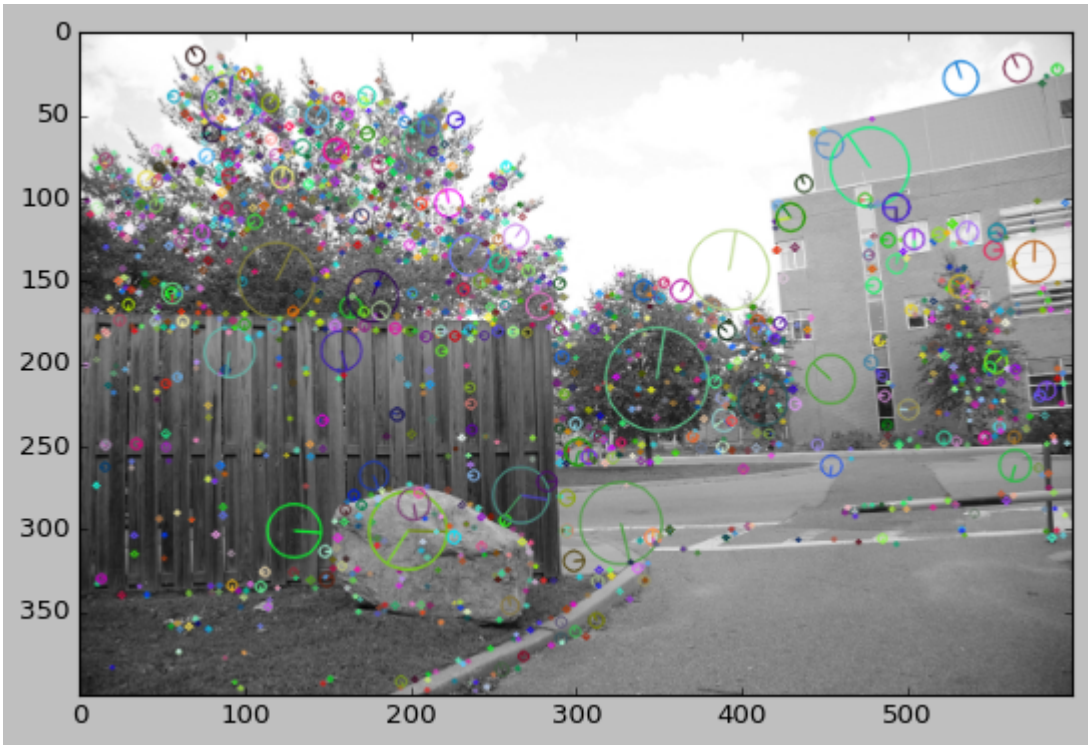
7.2.1. Use the OpenCV, Python, and MATLAB implementation of the SIFT or SURF operator to find interest points and establish correspondences between the images. In this case you can directly compare the feature vectors of interest points. You will match and align between different views of a scene with SIFT/SURF features. Discuss results and demonstrates the output of each method separately (Test on sl,sm,sr.jpg images).

In [ ]:
```
# construct a SIFT object
sift = cv2.SIFT_create()
```

in 1st phase, we find key points and description vectors:

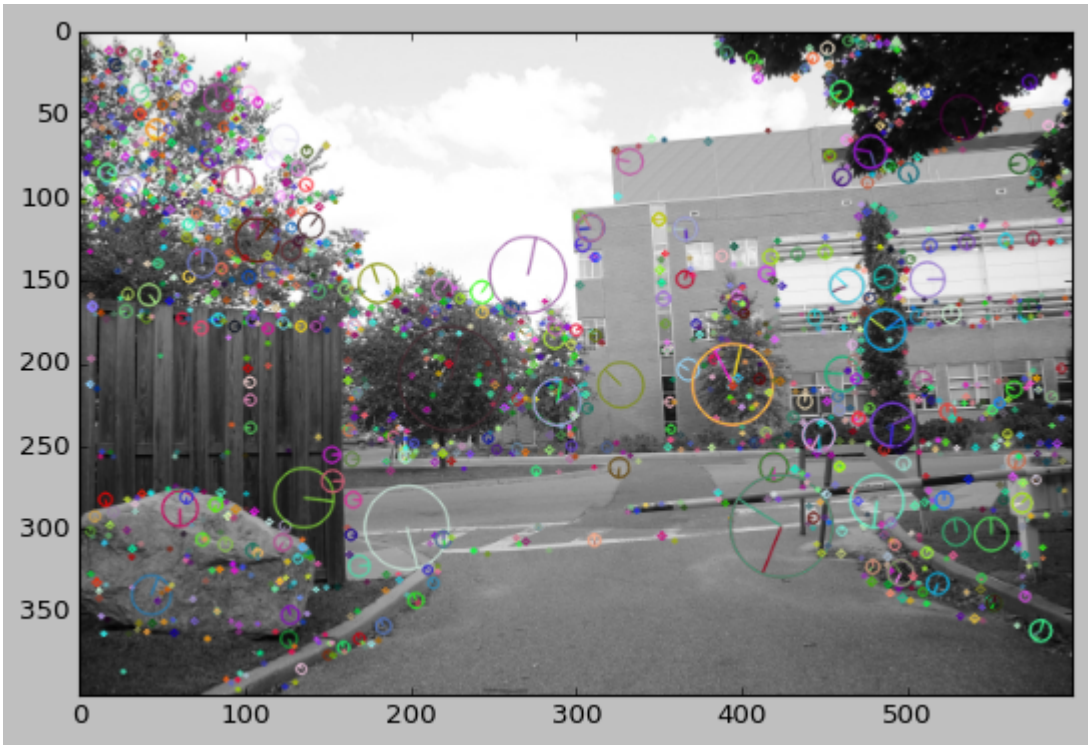
In [ ]:
```
sl = cv2.imread('sl.jpg')
sl = cv2.cvtColor(sl, cv2.COLOR_BGR2RGB)
sl_gray = cv2.cvtColor(sl, cv2.COLOR_BGR2GRAY)

# find keypoints and descriptors
sl_kp , sl_desc = sift.detectAndCompute(sl_gray.copy(),None)

# draws the small circles on the locations of keypoints
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS --> show keypoints orientation, also strc
img=cv2.drawKeypoints(sl_gray.copy(),sl_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158e8a02e60>
```



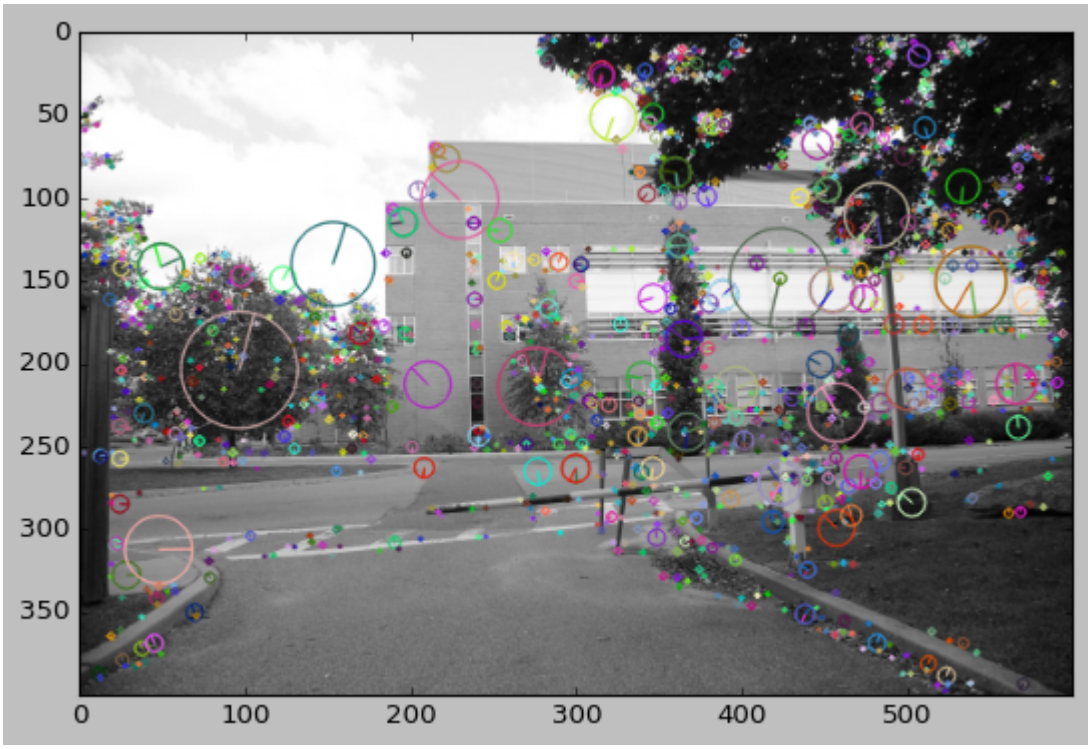
In [ ]:
```
sm = cv2.imread('sm.jpg')
sm = cv2.cvtColor(sm, cv2.COLOR_BGR2RGB)
sm_gray = cv2.cvtColor(sm, cv2.COLOR_BGR2GRAY)

sm_kp, sm_desc = sift.detectAndCompute(sm_gray.copy(),None)

img=cv2.drawKeypoints(sm_gray.copy(),sm_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158e8b399f0>
```



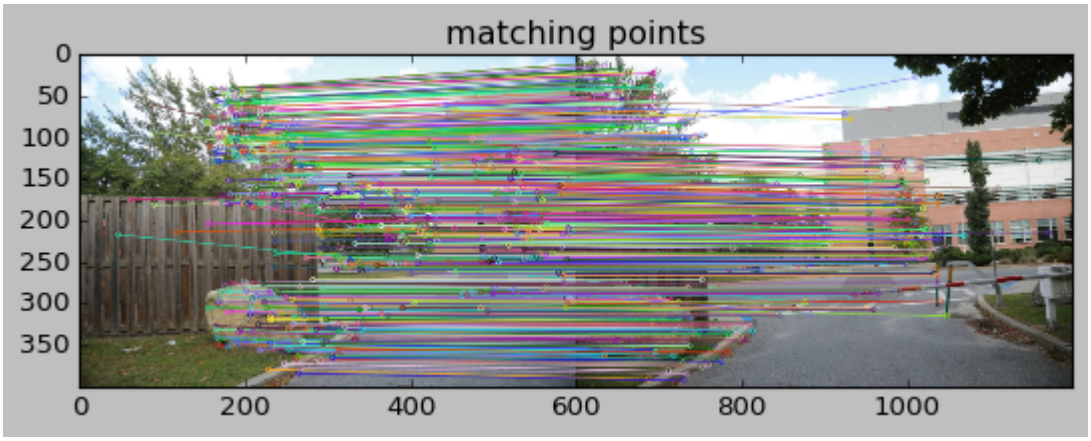
In [ ]:
```
sr = cv2.imread('sr.jpg')
sr = cv2.cvtColor(sr, cv2.COLOR_BGR2RGB)
sr_gray = cv2.cvtColor(sr, cv2.COLOR_BGR2GRAY)

sr_kp , sr_desc = sift.detectAndCompute(sr_gray.copy(),None)

img=cv2.drawKeypoints(sr_gray.copy(),sr_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158e8bd9210>
```



in 2nd phase, we stitch each 2 images of 3 provided images:

In [ ]:
```
# find similar interest points of sl and sm
good_matches , matches1to2 = BruteForceMatcher(sl_desc,sm_desc)


matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + sm_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and sm, show them on image
sl_sm = cv2.drawMatchesKnn(sl.copy(),sl_kp,sm.copy(),sm_kp,matches1to2,None,flags=cv2.
plt.imshow(sl_sm)
plt.title('matching points')
plt.show()
```



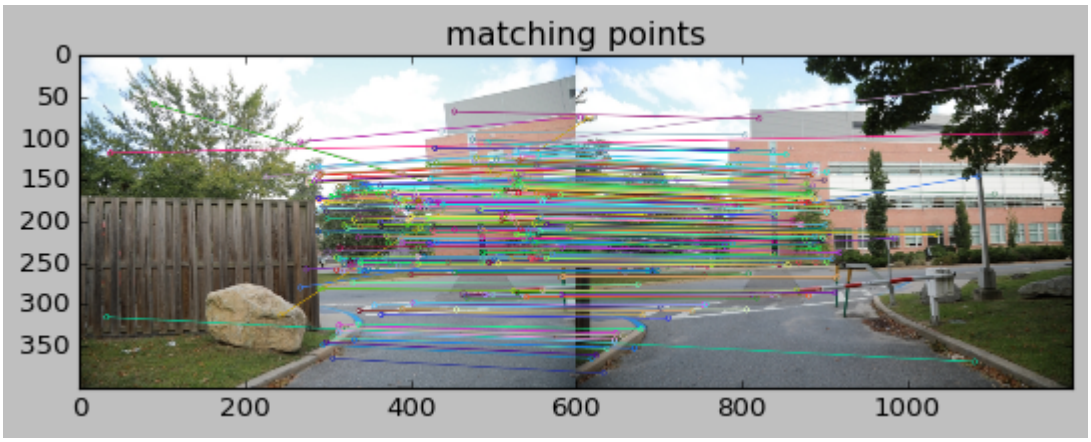
In [ ]:
```
# compute similarity percent of sl and sm
print(howSimilar(sl_kp,sm_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sl,sm,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
35.55291319857313
inliers/matches: 197/598
stiching image ...
  0%|          | 0/458 [00:00<?, ?it/s]
```

In [ ]:
```
# save result
plt.imsave("sl_sm_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

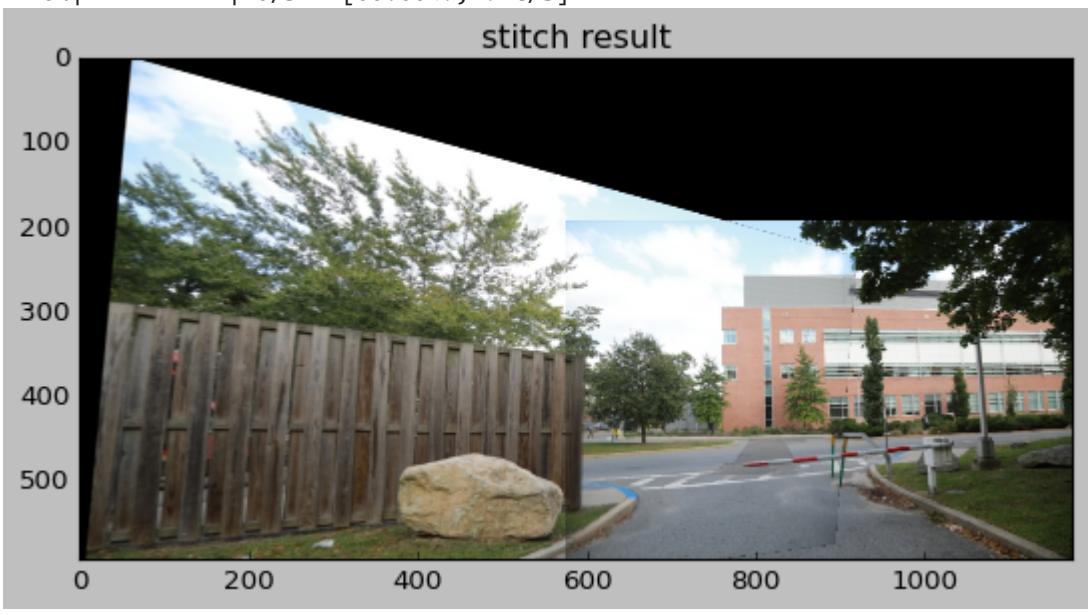
In [ ]:
```
# find similar interest points of sl and sr
good_matches , matches1to2 = BruteForceMatcher(sl_desc,sr_desc)

matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + sr_kp[pair.trainIdx].pt))
```

```
matches = np.array(matches)

# match similar interest points of sl and sr, show them on image
sl_sr = cv2.drawMatchesKnn(sl.copy(),sl_kp,sr.copy(),sr_kp,matches1to2,None,flags=cv2.
plt.imshow(sl_sr)
plt.title('matching points')
plt.show()
```



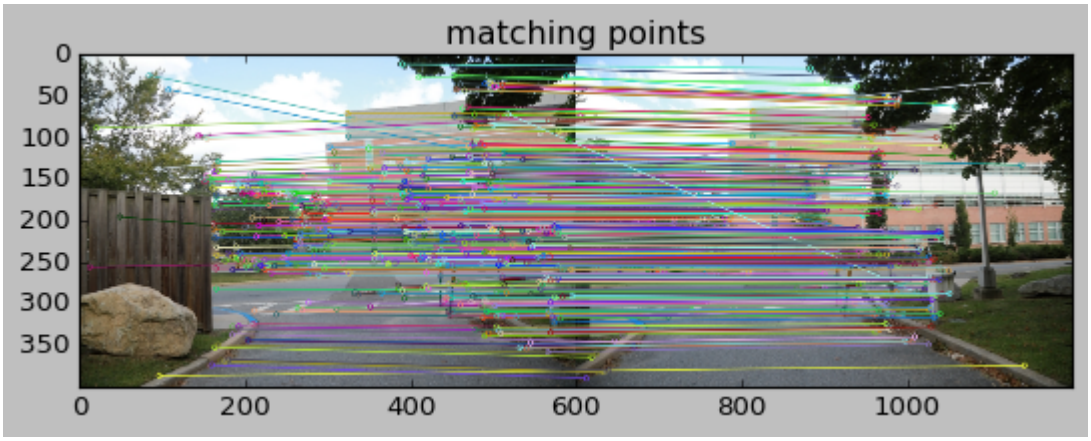
In [ ]:
```
# compute similarity percent of sl and sr
print(howSimilar(sl_kp,sr_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sl,sr,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
17.25219573400251
inliers/matches: 115/275
stiching image ...
  0%|          | 0/592 [00:00<?, ?it/s]
```



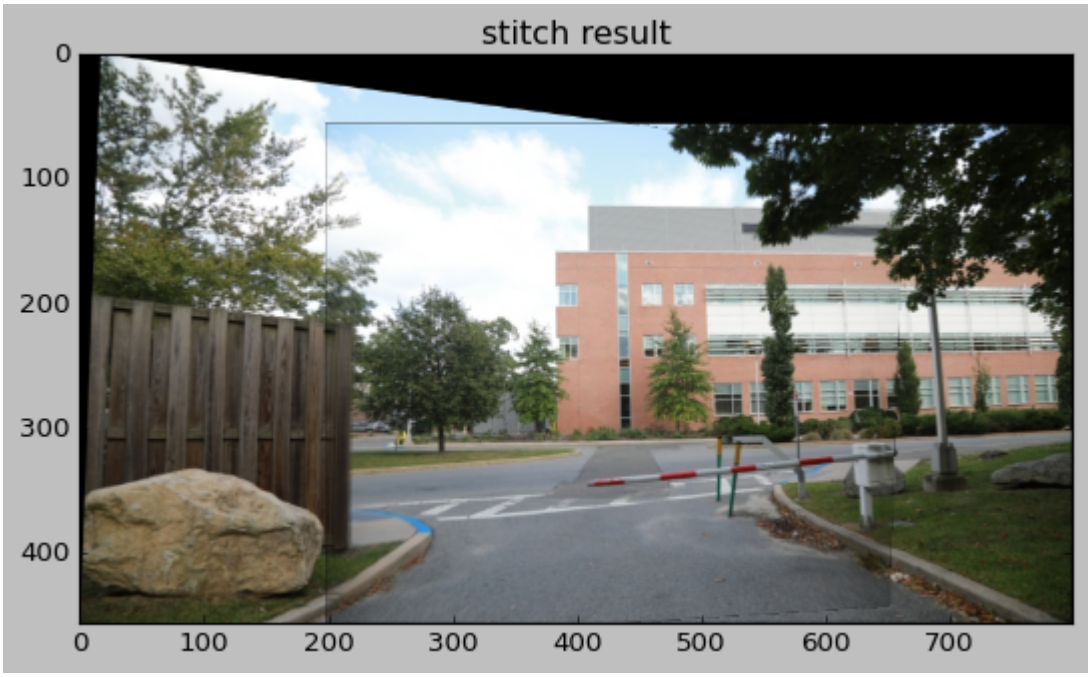
In [ ]:
```
plt.imsave("sl_sr_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

In [ ]:
```
# find similar interest points of sm and sr
good_matches , matches1to2 = BruteForceMatcher(sm_desc,sr_desc)


matches = []
for pair in good_matches:
    matches.append(list(sm_kp[pair.queryIdx].pt + sr_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and sm, show them on image
sm_sr = cv2.drawMatchesKnn(sm.copy(),sm_kp,sr.copy(),sr_kp,matches1to2,None,flags=cv2.
plt.imshow(sm_sr)
plt.title('matching points')
plt.show()
```



In [ ]:
```
# compute similarity percent of sm and sr
print(howSimilar(sm_kp,sr_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sm,sr,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
33.31242158092848
inliers/matches: 231/531
stiching image ...
  0%|          | 0/456 [00:00<?, ?it/s]
```

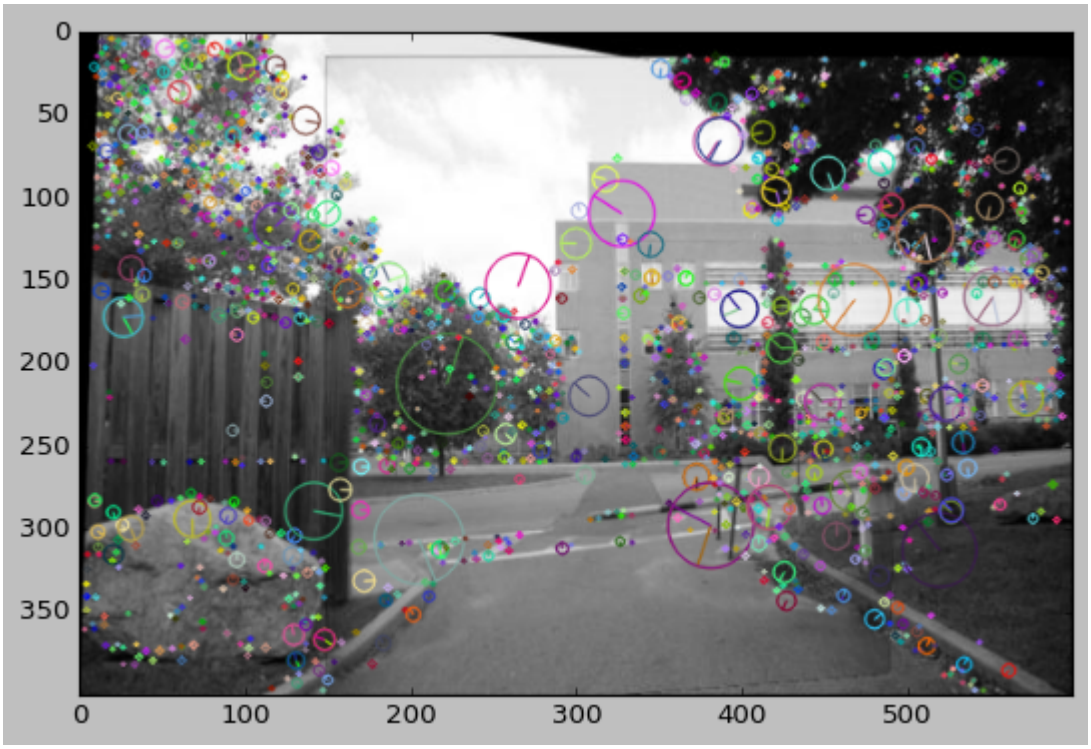


In [ ]:
```
plt.imsave("sm_sr_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

in last phase, we make a full view:

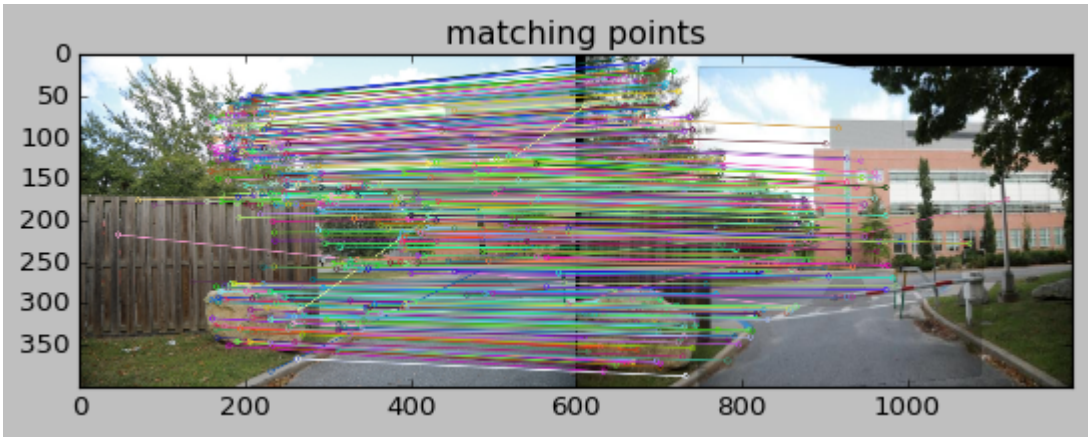
In [ ]:
```
result = cv2.imread('sm_sr_combined.jpg')
result = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
result_gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)

# find keypoints and descriptors
result_kp , result_desc = sift.detectAndCompute(result_gray.copy(),None)

# draws the small circles on the locations of keypoints
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS --> show keypoints orientation, also stro
img=cv2.drawKeypoints(result_gray.copy(),result_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_D

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158ea08ac80>
```

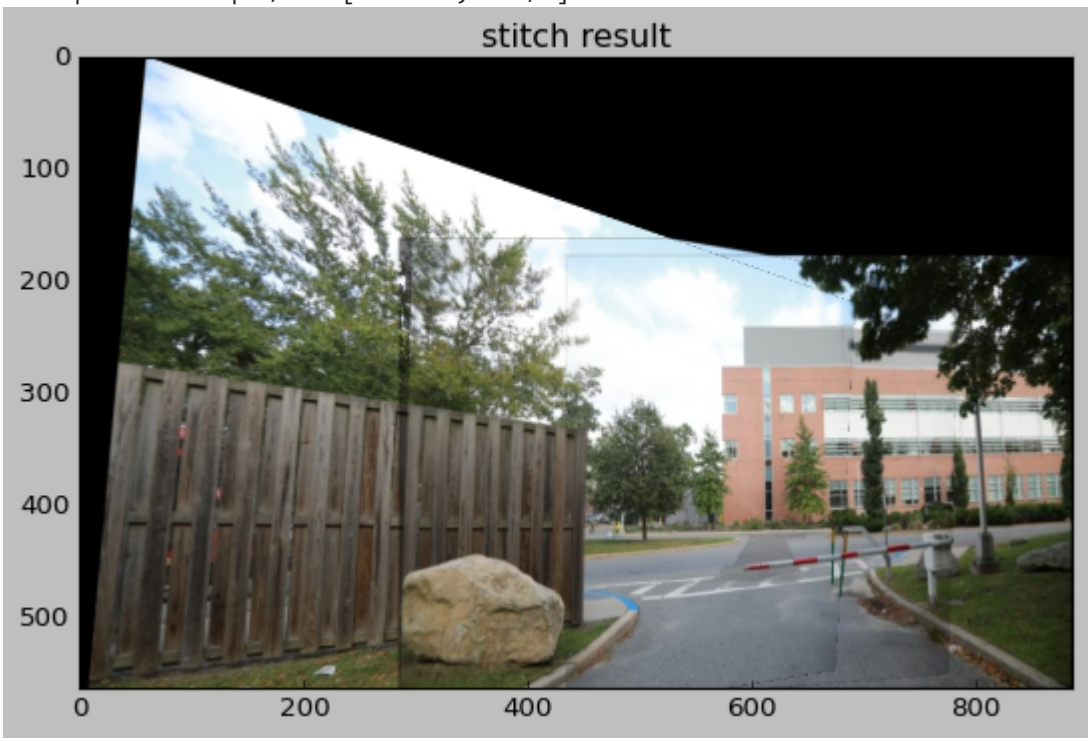


In [ ]:

```
# find similar interest points of sl and last step result
good_matches , matches1to2 = BruteForceMatcher(sl_desc,result_desc)


matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + result_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and result, show them on image
sl_result = cv2.drawMatchesKnn(sl.copy(),sl_kp,result.copy(),result_kp,matches1to2,Nor
plt.imshow(sl_result)
plt.title('matching points')
plt.show()

# compute similarity percent of sl and result
print(howSimilar(sl_kp,result_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

complete = stitch(sl,result,H)
plt.imshow(complete)
plt.title('stitch result')
plt.show()
```




```
24.37574316290131
inliers/matches: 124/410
stiching image ...
  0%|          | 0/561 [00:00<?, ?it/s]
```



In [ ]:
```
plt.imsave("complete.jpg",complete[130: , 20:])
```

I used this links for implements:

https://pyimagesearch.com/2016/01/11/opencv-panorama-stitching/

https://gist.github.com/tigercosmos/90a5664a3b698dc9a4c72bc0fcbd21f4

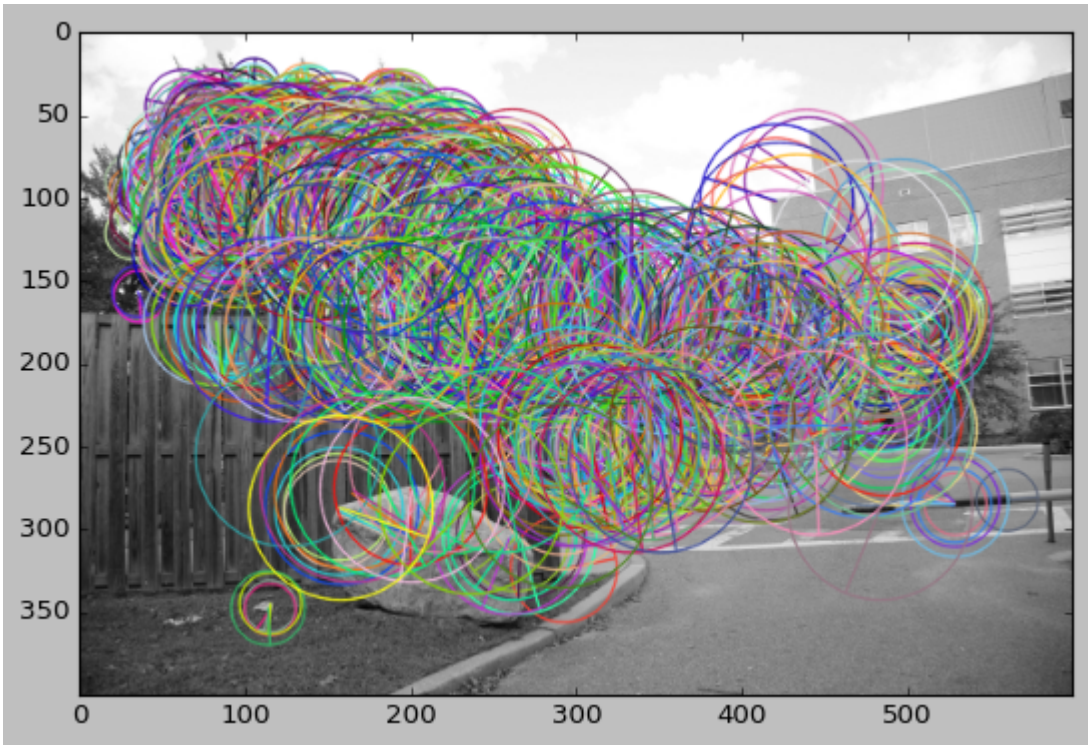
ORB in opencv

In [ ]:
```
# construct a ORB object
fast = cv2.ORB_create()
```

In [ ]:
```
sl = cv2.imread('sl.jpg')
sl = cv2.cvtColor(sl, cv2.COLOR_BGR2RGB)
sl_gray = cv2.cvtColor(sl, cv2.COLOR_BGR2GRAY)
```

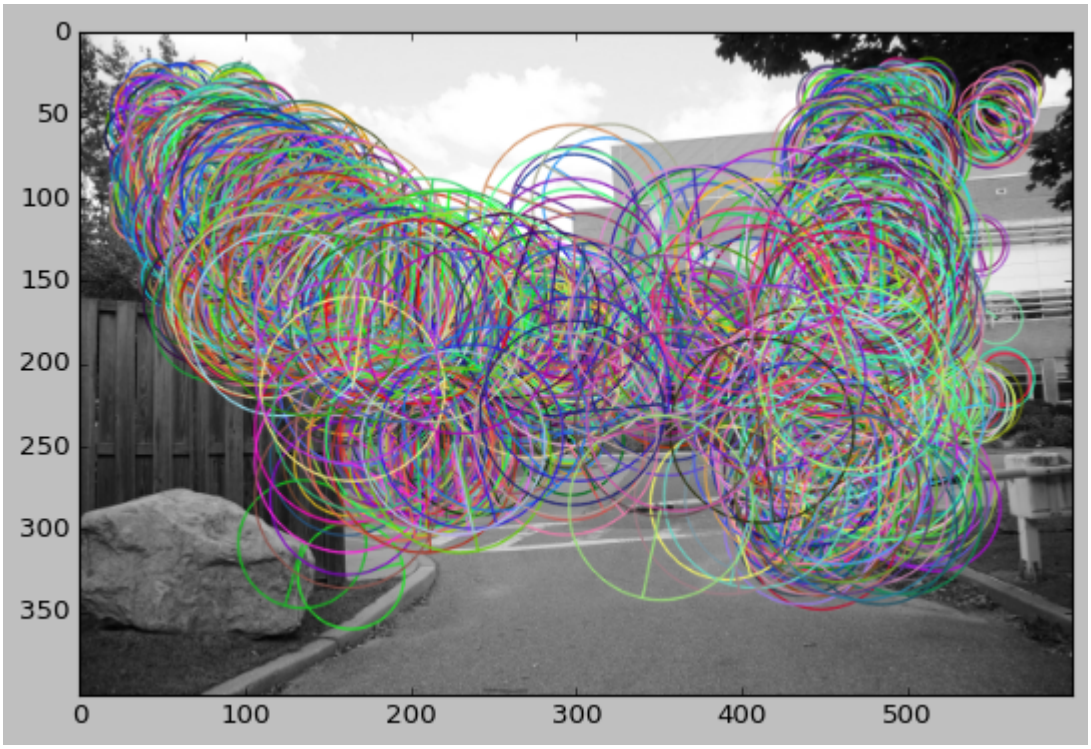
```
# find keypoints and descriptors
sl_kp , sl_desc = fast.detectAndCompute(sl_gray.copy(),None)

# draws the small circles on the locations of keypoints
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS --> show keypoints orientation, also stro
img=cv2.drawKeypoints(sl_gray.copy(),sl_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158ef5bb2e0>
```



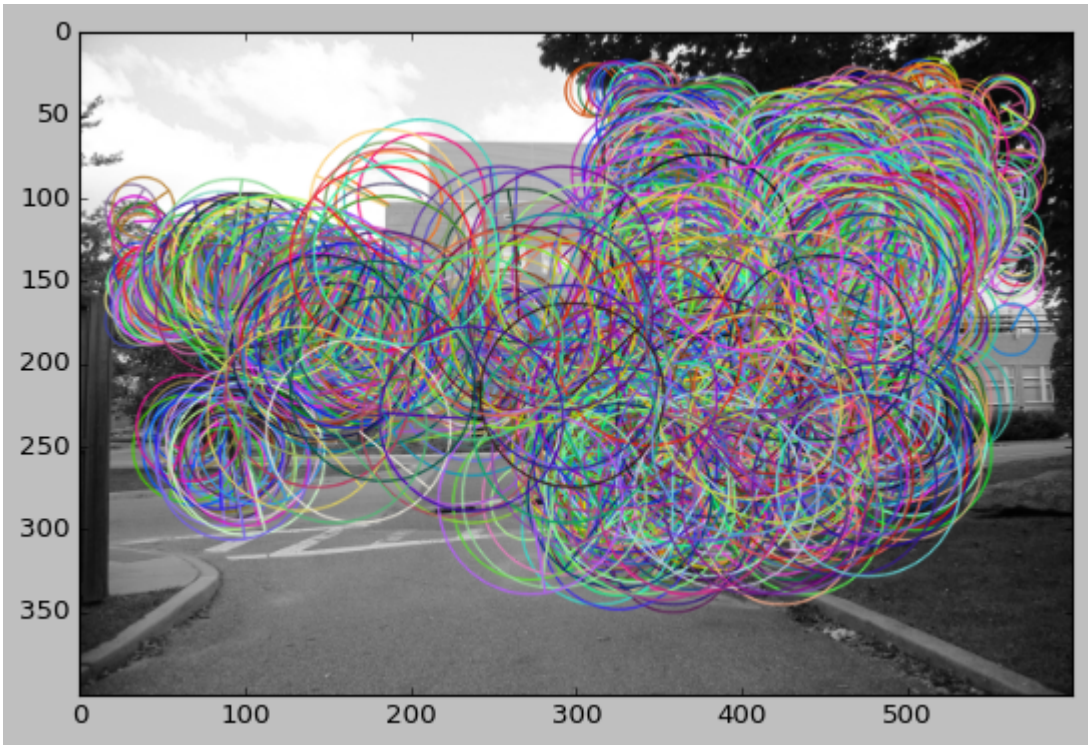
In [ ]:
```
sm = cv2.imread('sm.jpg')
sm = cv2.cvtColor(sm, cv2.COLOR_BGR2RGB)
sm_gray = cv2.cvtColor(sm, cv2.COLOR_BGR2GRAY)

sm_kp, sm_desc = fast.detectAndCompute(sm_gray.copy(),None)

img=cv2.drawKeypoints(sm_gray.copy(),sm_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158e9e36620>
```

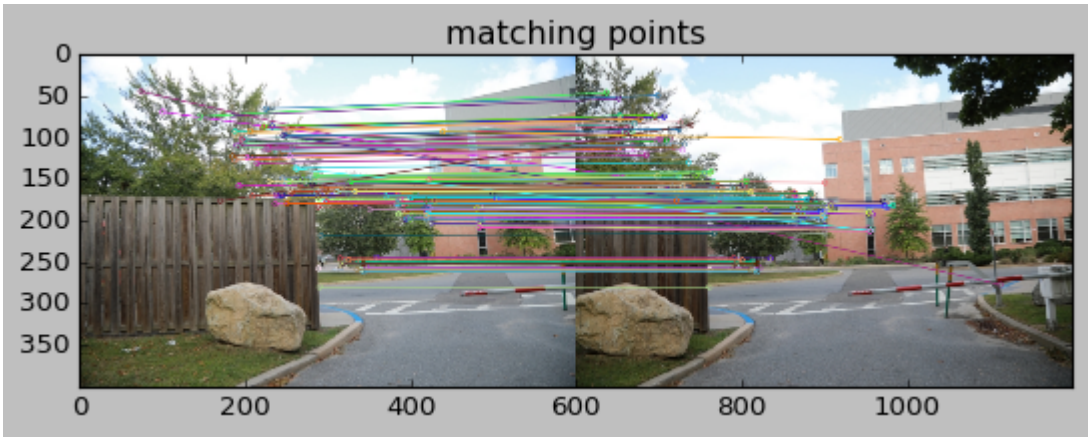


In [ ]:
```
sr = cv2.imread('sr.jpg')
sr = cv2.cvtColor(sr, cv2.COLOR_BGR2RGB)
sr_gray = cv2.cvtColor(sr, cv2.COLOR_BGR2GRAY)

sr_kp , sr_desc = fast.detectAndCompute(sr_gray.copy(),None)

img=cv2.drawKeypoints(sr_gray.copy(),sr_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x158e9140b20>
```

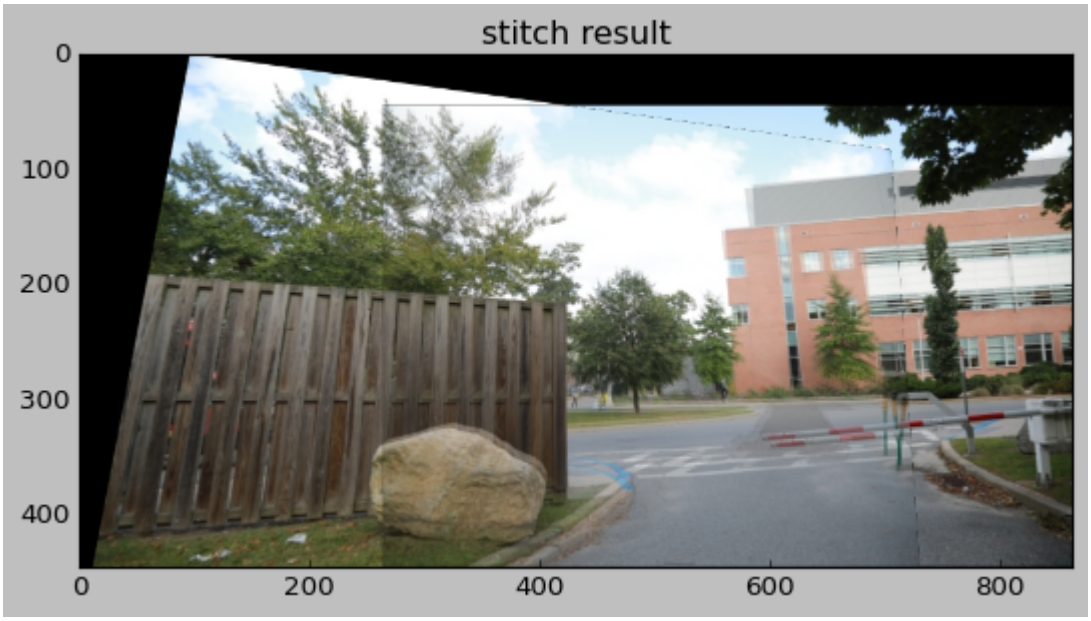


In [ ]:
```
# find similar interest points of sl and sm
good_matches , matches1to2 = BruteForceMatcher(sl_desc,sm_desc)


matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + sm_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and sm, show them on image
sl_sm = cv2.drawMatchesKnn(sl.copy(),sl_kp,sm.copy(),sm_kp,matches1to2,None,flags=cv2.
plt.imshow(sl_sm)
plt.title('matching points')
plt.show()
```



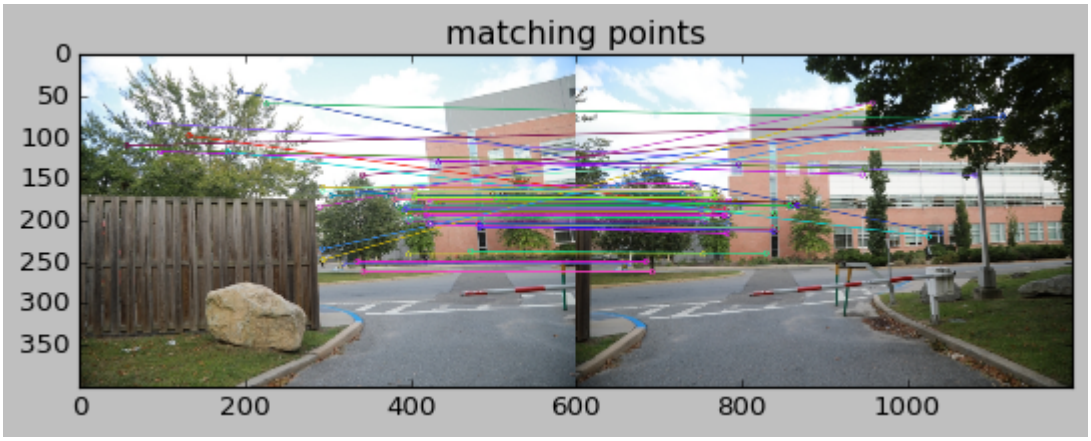
In [ ]:
```
# compute similarity percent of sl and sm
print(howSimilar(sl_kp,sm_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sl,sm,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
9.319899244332493
inliers/matches: 29/185
stiching image ...
  0%|          | 0/445 [00:00<?, ?it/s]
```



In [ ]:
```
# save result
plt.imsave("fast_sl_sm_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

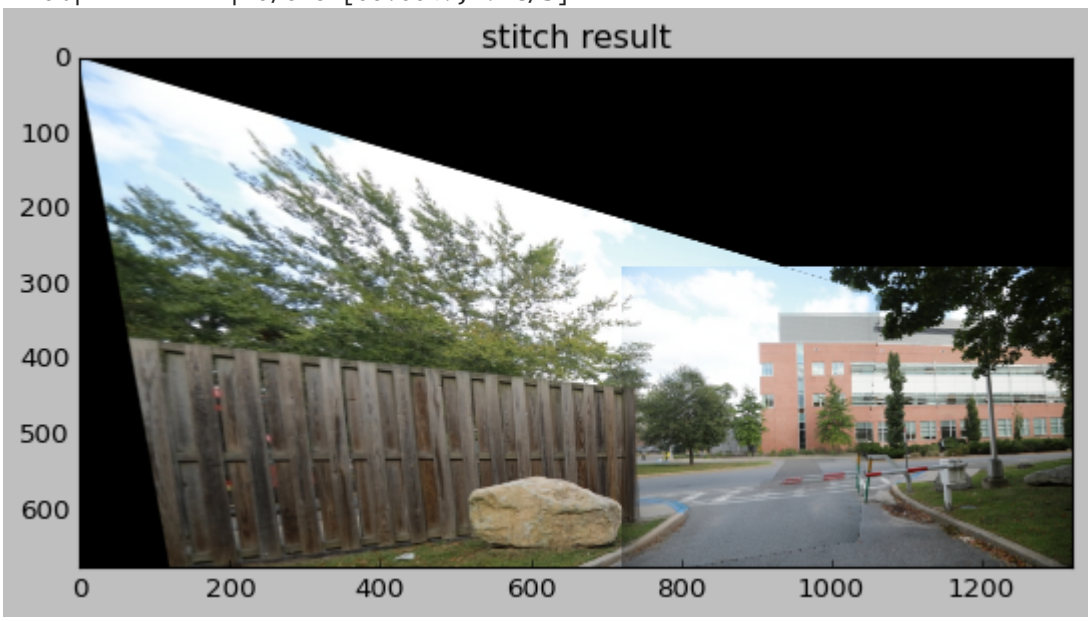
In [ ]:
```
# find similar interest points of sl and sr
good_matches , matches1to2 = BruteForceMatcher(sl_desc,sr_desc)

matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + sr_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and sr, show them on image
sl_sr = cv2.drawMatchesKnn(sl.copy(),sl_kp,sr.copy(),sr_kp,matches1to2,None,flags=cv2.
plt.imshow(sl_sr)
plt.title('matching points')
plt.show()
```



In [ ]:
```
# compute similarity percent of sl and sr
print(howSimilar(sl_kp,sr_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sl,sr,H)
```

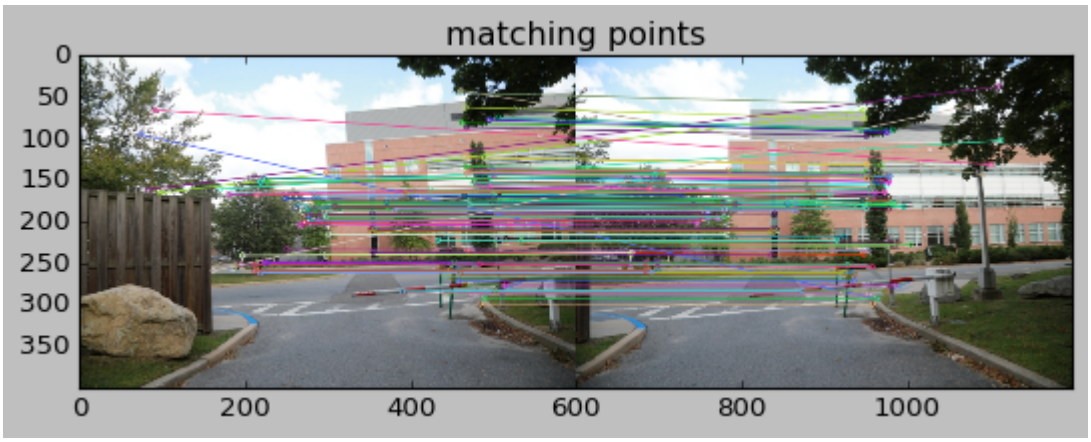
```
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
4.433249370277078
inliers/matches: 16/88
stiching image ...
  0%|          | 0/676 [00:00<?, ?it/s]
```



In [ ]:
```
plt.imsave("fast_sl_sr_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

In [ ]:
```
# find similar interest points of sm and sr
good_matches , matches1to2 = BruteForceMatcher(sm_desc,sr_desc)


matches = []
for pair in good_matches:
    matches.append(list(sm_kp[pair.queryIdx].pt + sr_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and sm, show them on image
sm_sr = cv2.drawMatchesKnn(sm.copy(),sm_kp,sr.copy(),sr_kp,matches1to2,None,flags=cv2.
plt.imshow(sm_sr)
plt.title('matching points')
plt.show()
```



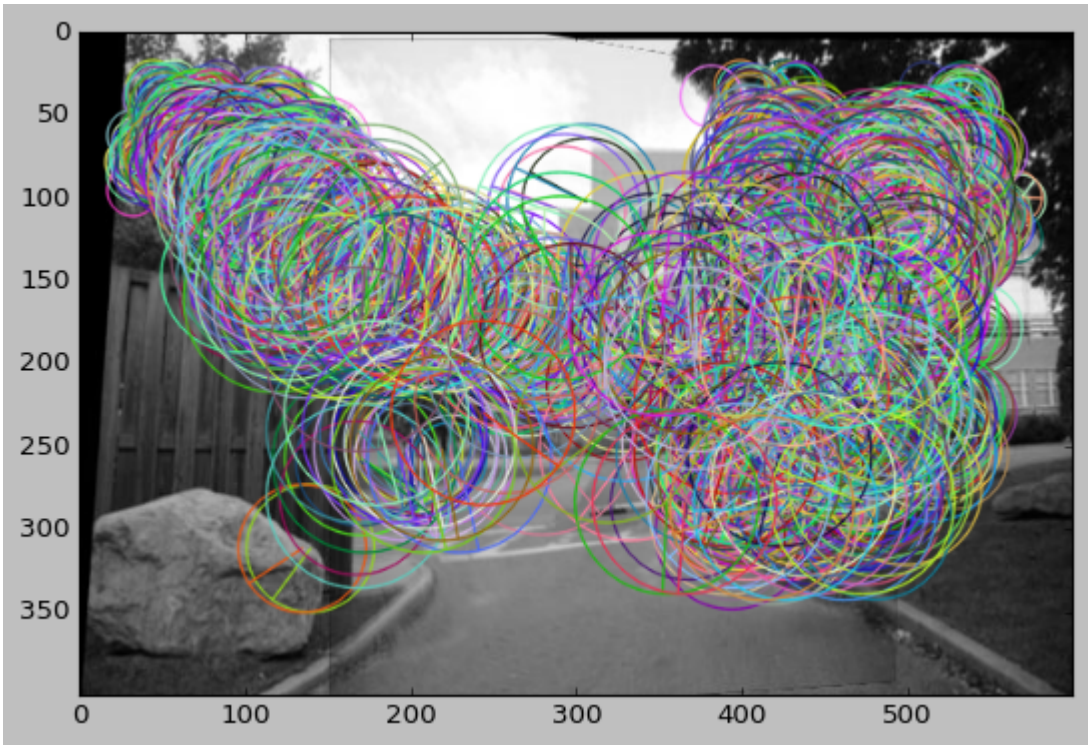
In [ ]:
```
# compute similarity percent of sm and sr
print(howSimilar(sm_kp,sr_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(sm,sr,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()
```

```
8.7
inliers/matches: 40/174
stiching image ...
  0%|          | 0/444 [00:00<?, ?it/s]
```

In [ ]:
```
plt.imsave("fast_sm_sr_combined.jpg",cv2.resize(result[40: , :],(600,400)))
```

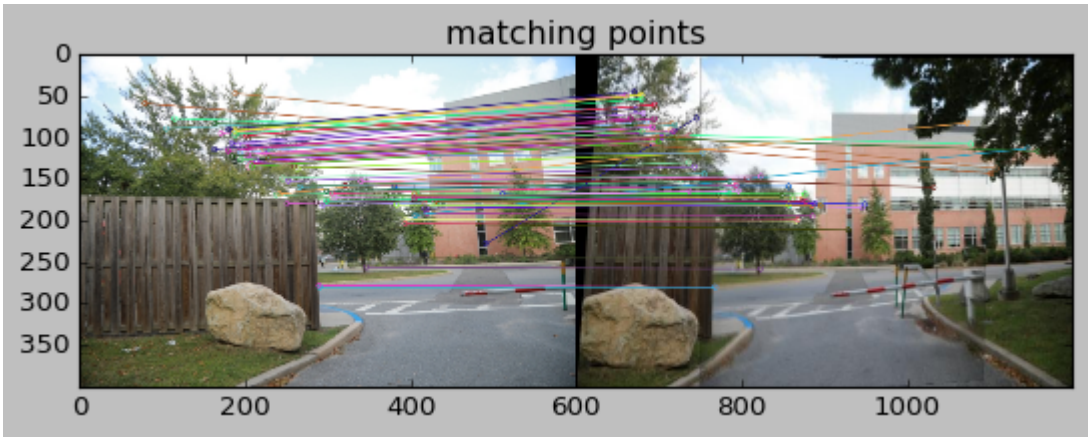
In [ ]:
```
result = cv2.imread('fast_sm_sr_combined.jpg')
result = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)
result_gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)

# find keypoints and descriptors
result_kp , result_desc = fast.detectAndCompute(result_gray.copy(),None)

# draws the small circles on the locations of keypoints
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS --> show keypoints orientation, also strc
img=cv2.drawKeypoints(result_gray.copy(),result_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_D

plt.imshow(img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x15882d9e530>
```

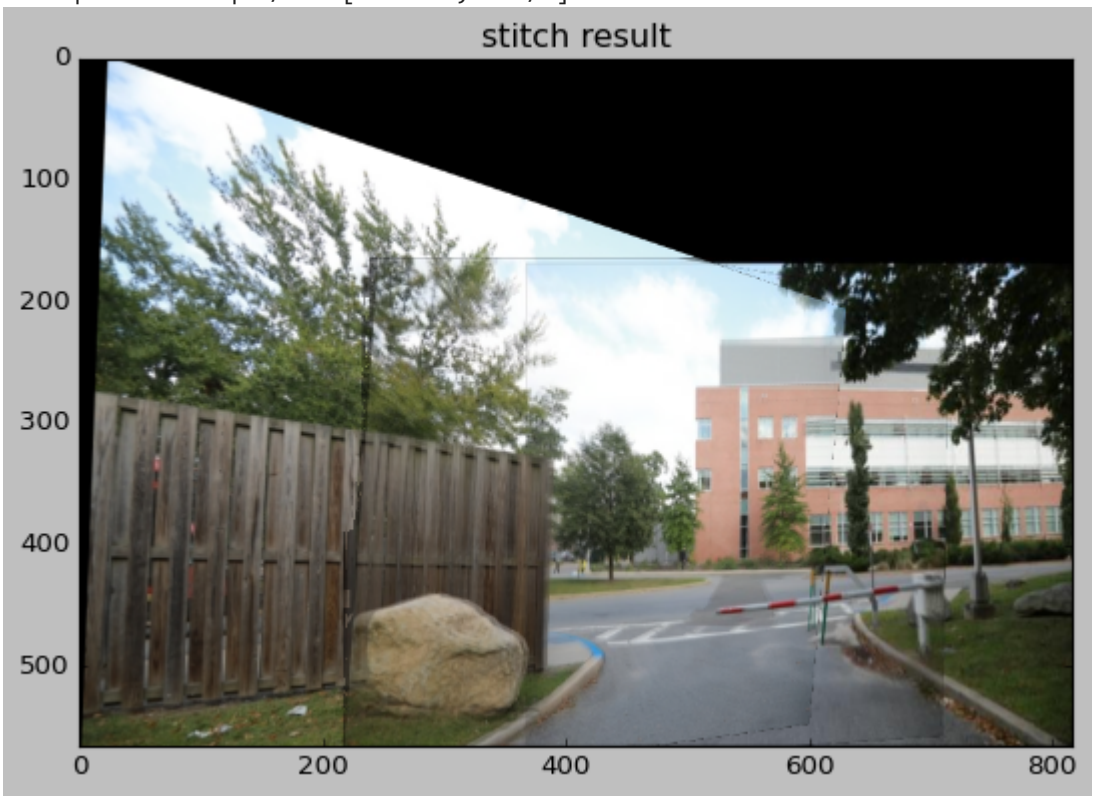


In [ ]:

```
# find similar interest points of sl and last step result
good_matches , matches1to2 = BruteForceMatcher(sl_desc,result_desc)


matches = []
for pair in good_matches:
    matches.append(list(sl_kp[pair.queryIdx].pt + result_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of sl and result, show them on image
sl_result = cv2.drawMatchesKnn(sl.copy(),sl_kp,result.copy(),result_kp,matches1to2,Nor
plt.imshow(sl_result)
plt.title('matching points')
plt.show()

# compute similarity percent of sl and result
print(howSimilar(sl_kp,result_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

complete = stitch(sl,result,H)
plt.imshow(complete)
plt.title('stitch result')
plt.show()
```




```
4.231738035264484
inliers/matches: 13/84
stiching image ...
  0%|          | 0/565 [00:00<?, ?it/s]
```

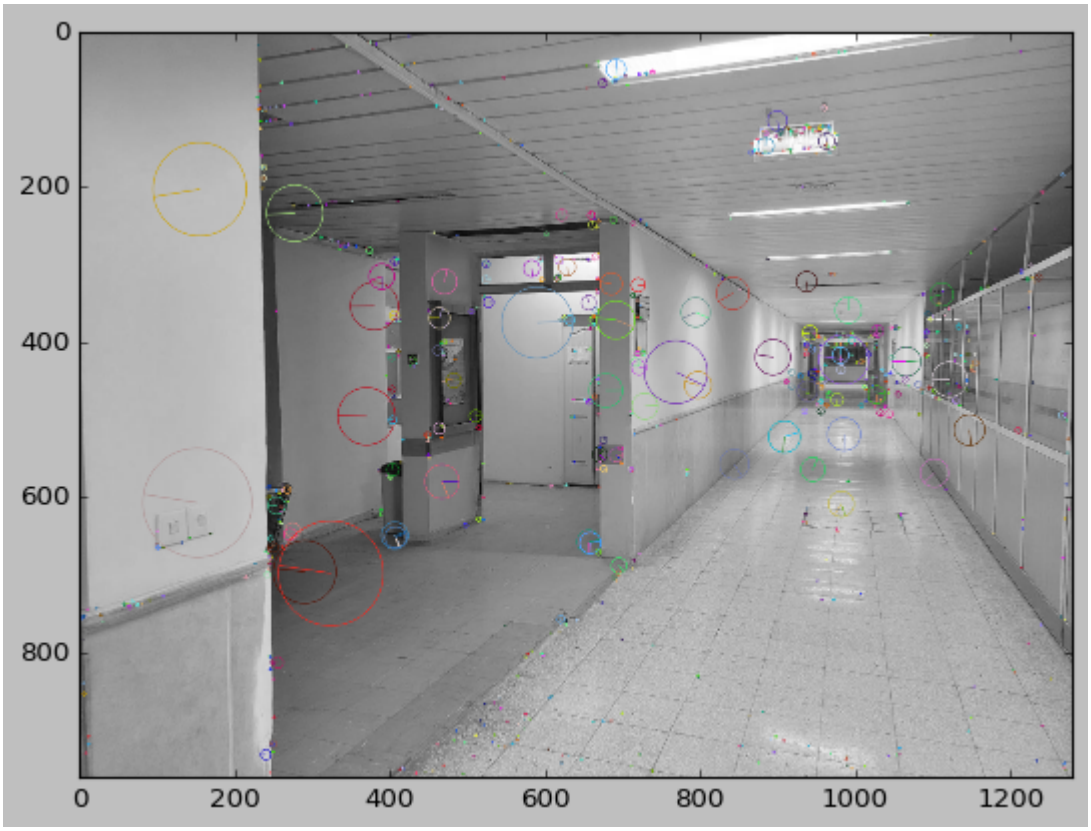


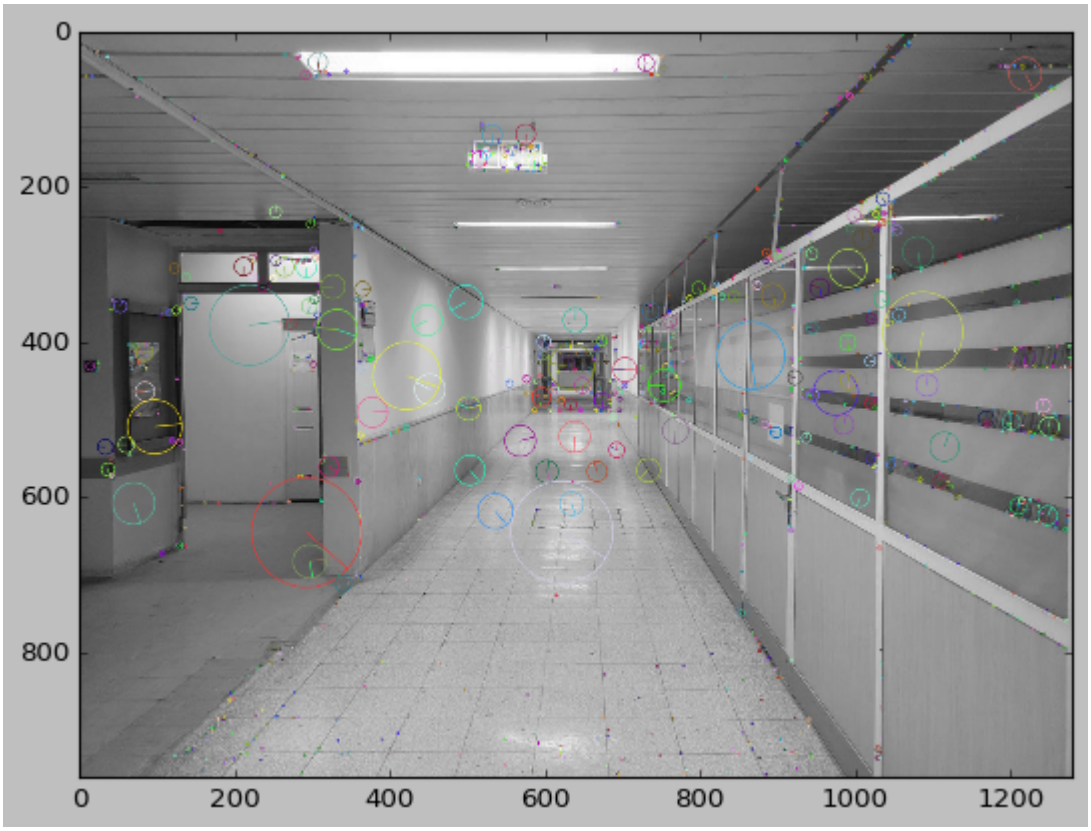
In [ ]:
```
plt.imsave("fast_complete.jpg",complete[130: , 20:])
```

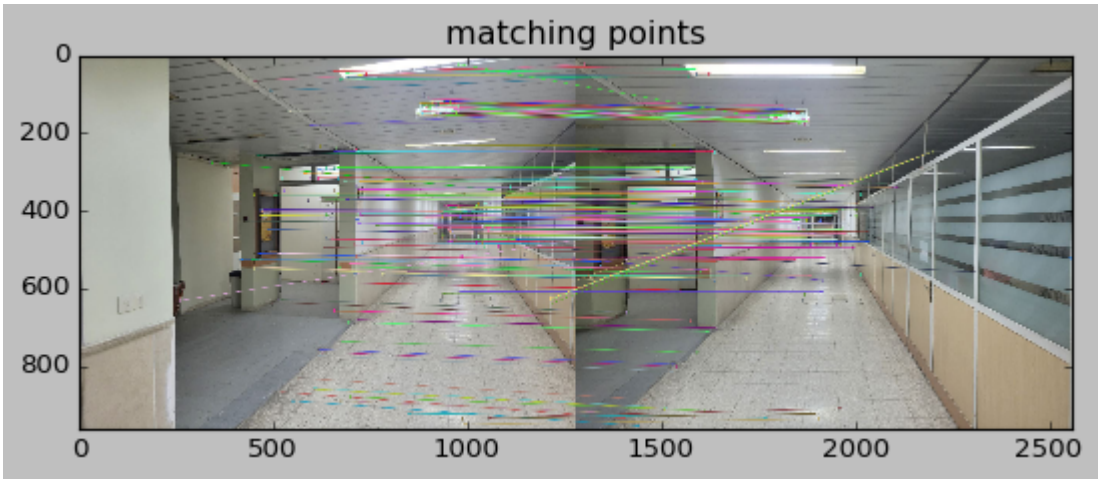
7.2.2. Do these steps by images that are taken with your own camera as well.

In [ ]:
```
# construct a SIFT object
sift = cv2.SIFT_create()

ApaL = cv2.imread('ApaL.jpg')
ApaL = cv2.cvtColor(ApaL, cv2.COLOR_BGR2RGB)
ApaL_gray = cv2.cvtColor(ApaL, cv2.COLOR_BGR2GRAY)

# find keypoints and descriptors
ApaL_kp , ApaL_desc = sift.detectAndCompute(ApaL_gray.copy(),None)

# draws the small circles on the locations of keypoints
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS --> show keypoints orientation, also strc
```

```
img=cv2.drawKeypoints(ApaL_gray.copy(),ApaL_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_

plt.imshow(img)
plt.show()

ApaR = cv2.imread('ApaR.jpg')
ApaR = cv2.cvtColor(ApaR, cv2.COLOR_BGR2RGB)
ApaR_gray = cv2.cvtColor(ApaR, cv2.COLOR_BGR2GRAY)

ApaR_kp, ApaR_desc = sift.detectAndCompute(ApaR_gray.copy(),None)

img=cv2.drawKeypoints(ApaR_gray.copy(),ApaR_kp,None,flags=cv2.DRAW_MATCHES_FLAGS_DRAW_

plt.imshow(img)
plt.show()

# find similar interest points of ApaL and ApaR
good_matches , matches1to2 = BruteForceMatcher(ApaL_desc,ApaR_desc)


matches = []
for pair in good_matches:
    matches.append(list(ApaL_kp[pair.queryIdx].pt + ApaR_kp[pair.trainIdx].pt))

matches = np.array(matches)

# match similar interest points of ApaL and ApaR, show them on image
ApaL_ApaR = cv2.drawMatchesKnn(ApaL.copy(),ApaL_kp,ApaR.copy(),ApaR_kp,matches1to2,Nor
plt.imshow(ApaL_ApaR)
plt.title('matching points')
plt.show()

# compute similarity percent of ApaL and ApaR
print(howSimilar(ApaL_kp,ApaR_kp,good_matches))

# compute H matrx and best match points
inliers, H = ransac(matches, 0.2, 2000)

result = stitch(ApaL,ApaR,H)
plt.imshow(result)
plt.title('stitch result')
plt.show()

# save result
plt.imsave("ApaL_ApaR_combined.jpg",cv2.resize(result[40: , :],(1000,600)))
```


0
200
400
600
800
0
200
400
600
800
1000
1200


0
200
400
600
800
0
200
400
600
800
1000
1200




```
25.747863247863243
inliers/matches: 23/241
stiching image ...
  0%|          | 0/1123 [00:00<?, ?it/s]
```